Naval Research Laboratory
Stennis Space Center, MS 39529-5004

NRL/MR/7441--96-7718

Technical Review of the Vector Product Format Symbol Set Prototype

JERRY L. LANDRUM
KEVIN B. SHAW

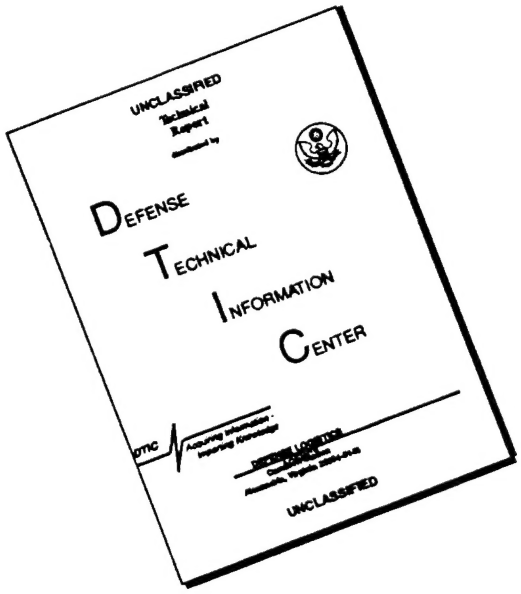
Mapping, Charting, and Geodesy Branch
Marine Geosciences Division

THOMAS A. FETTERER

Planning Systems Incorporated
Slidell, LA

April 12, 1996

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

REPORT DOCUMENTATION PAGE

Form Approved
OBM No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	April 12, 1996	Final

4. TITLE AND SUBTITLE

Technical Review of the Vector Product Format Symbol Set Prototype

5. FUNDING NUMBERS

Job Order No. 5745137A6
Program Element No. 0603704N
Project No. R1987
Task No. 300
Accession No. DN257086

6. AUTHOR(S)

Jerry L. Landrum, Kevin B. Shaw, and Thomas A. Fetterer*

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory
Marine Geosciences Division
Stennis Space Center, MS 39529-5004

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL/MR/7441--96-7718

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Oceanographer of the Navy
U.S. Naval Observatory
34th & Massachusetts Ave. NW
Washington DC 20392-1800

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

Planning Systems Incorporated, 115 Christian Lane, Slidell, LA 70458

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Vector Product Format Symbology (VPFS) is a standard structure for the organization of digital symbology to be used with Vector Product Format (VPF) products. The symbols it contains are designed to be compatible with a wide variety of applications which use VPF. VPFS supports a greatly enriched symbology by incorporating logical conditions that allow the association of symbology with feature attribute values. VPFS encompasses most, if not all, of the VPF products providing a standard and consistent symbolization of features across VPF product lines. As the use of multiple VPF products becomes common, this standard symbology set will greatly ease the conflicts between graphical representations of features in different databases. This review examines the appropriateness of the VPFS standard and specification in concept as well as the efficacy of VPFS as implemented in the prototype. It also presents suggestions for the improvement of the VPFS product design.

14. SUBJECT TERMS

requirements, MC&G data, mapping, DCW, ADRG, WVS

15. NUMBER OF PAGES

20

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	SAR

Contents

Technical Review of the Vector Product Format Symbol Set Prototype

1. Introduction

Vector Product Format Symbology (VPFS) is a standard structure for the organization of digital symbology to be used with Vector Product Format (VPF) products. The symbols it contains are designed to be compatible with a wide variety of applications which use VPF. VPFS supports a greatly enriched symbology by incorporating logical conditions that allow the association of symbology with feature attribute values. VPFS encompasses most, if not all, of the VPF products providing a standard and consistent symbolization of features across VPF product lines. As the use of multiple VPF products becomes common, this standard symbology set will greatly ease the conflicts between graphical representations of features in different databases.

This review examines the appropriateness of the VPFS standard and specification in concept as well as the efficacy of VPFS as implemented in the prototype. It also presents suggestions for the improvement of the VPFS product.

1.1 The Evaluation Package

The evaluation package consists of an ISO 9660 CDROM labeled "Symbology Prototype" and dated February 1995.

The CDROM has a directory named "docs" containing the following documents in several formats:

MIL-STD-2412, Vector Product Format Symbology Military Standard, herein referred to as the standard.

MIL-V-89045, Vector Product Format Military Specification, herein referred to as the specification.

MIL-HDBK-857, Vector Product Format Military Handbook, herein referred to as the handbook.

The CDROM also contains the prototype VPFS database in a directory named "symproto".

The following sample VPF data products are included: DFLIP, DNC01, UVMAP, VITD, VMAPLV0, VMAPLV1, VMAPLV2, and WVS.

The VPFVIEW software is included for DOS and Sun/UNIX along with installation software. The installation includes predefined views for the sample databases and the imported symbology database.

1.2 General Description of VPFS

The software application system used in the VPFS evaluation was provided on the VPFS CDROM and consisted of the documents, the VPFVIEW program, sample VPF databases, sample views, and the VPFS. Where a weakness or limitation appears in the application system, it is not always obvious which part of the system may be at fault.

VPFS is presented as a VPF database with a Database Header Table (DHT). A Library Header Table (LHT) is required by the VPFS standard, but was not in the prototype. The prototype looks like a VPF database but is not an actual VPF database in format or in content. The prototype contains the directories cond, desc, and graphics explained below:

cond- contains symbol condition tables, one table for each feature code, named by the feature code. The rows in these tables contain feature attribute expressions (conditions) and row pointers into the four symbol description tables. Four pointers are required only if the feature code is used within the union of all of the VPF data products to represent all four of the feature classes (point, line, area, and text). Most of these tables contain only 1 row of data with no symbol conditions, as most feature codes are appropriately symbolized in only one way. If the table contains more than one row, then the symbol conditions are compared to the feature's attributes to determine how best to symbolize this particular feature.

desc- contains symbol description tables, one table for each feature type (point, line, area, text) with corresponding names. These tables contain the names of the graphics files along with some additional symbol attributes.

graphics- contains graphics symbols in Computer Graphics Metafile (CGM) clear text, CGM binary, and/or X window Bit Map (XBM). Point symbol graphic files contain the CGM commands to draw a point symbol. Line symbol graphic files presumably contain some type of line pattern. Area symbol graphic files contain a pattern. Text symbology graphic files are not a part of the database, but presumably their descriptions would be analogous to font specifications.

To symbolize a particular feature, application software would:

1) locate and read the symbol condition table file having the same name as the feature's feature code. If the condition table contains only 1 row of data then the appropriate pointer is followed into the appropriate symbol description table. Which symbol description table pointer to follow is determined by the features class. If the table contains more than one row, then the symbol conditions are compared to the feature's attributes to determine how best to symbolize this particular feature.

2) Follow the row pointer into the feature description table to get the symbol attributes and the graphic file name.

3) Set drawing attributes based on the symbol attributes

4) Open and read the symbol from the graphic file.

5) If the symbol is a point or text feature, draw the point or text symbol. Otherwise if the symbol is a line or area feature, set the line or fill pattern and then draw the feature.

A more detailed walk through is provided by Appendix A.

Please note that this not the way that VPFVIEW handles the symbology. In VPFVIEW, the symbols are imported as a separate operation. Import is the only time that VPFVIEW accesses the VPFS. Symbology assignment is totally controlled by the user during the view configuration and is assigned at the theme level. None of the condition tables in the VPFS database are currently consulted.

1.3 Problems with the Evaluation Package

The evaluation package is complete in terms of the new product to be evaluated, evaluation software, and sample data, but it has some deficiencies in terms of consistency and in functionality.

The prototype differs significantly from the documents (tables are not VPF tables, table contents differ significantly from the documents). Where there is a conflict, this review gives preference to the documents.

The evaluation software (VPFVIEW) can import symbology from the CGM files, but it appears to provide only query level symbology, ignoring the part of the VPFS design relating to the use of the symbol conditions to provide feature level symbology.

2. VPFS Prototype Evaluation

2.1 Appropriateness of VPF to the Symbology Task

VPF is uniquely designed for the storage and retrieval of spatial data. As such the VPF format does not always lend itself to the inclusion of nonspatial data types. Indeed, the VPF standard specifically excludes nonspatial data.

Although section 4.3 in draft MIL-STD-2412 specifies that VPFS will use the structure and organization defined within the Vector Product Standard specification document (MIL-STD-2407), large discrepancies in data organization exist between VPF and VPFS. The VPF specification (MIL-STD-2407) clearly delimits the method of defining data in a VPF database. Some excerpts include:

Section 4.2.g (VPF Characteristics) states that "VPF allows application software to read data directly from the storage medium without prior conversion to another format. VPF uses tables and indexes that permit direct access by spatial location and thematic content."

Section 5.2.1 (Data Organization) asserts that "VPF uses only three types of files: directories, tables, and indexes." Section 5.2.1.2 refines the point further with the statement that "... the table is the organizational structure for all data content. All tables in a VPF database share a common basic structure; this structure... is mandatory for all VPF tables."

Section 5.2.1.3 (VPF Table Components) describes a VPF table as consisting of "... the following parts: a table header, a row identifier, and the table contents. ... contents in VPF tables are organized into rows and columns."

Section 5.2.2.1 (Primitives) specifies four types of primitives in VPF: nodes, edges, faces, and text. It further states that "All primitives except text can be linked to each other by topological relationships..."

As a CGM file does not mirror the structure of a VPF table, cannot be linked to any other VPF primitive table topologically, and is not accessible by spatial location, it cannot be considered a valid addition to a VPF database if strict adherence to MIL-STD-2407 is desired.

As mentioned above, MIL-STD-2407 (VPF) defines only three types of entities in a VPF database: directories, tables, and indexes. A CGM format file, although a standard format in its own right, does not meet the requirements of any of these three types. Two of the overriding characteristics of a VPF database,

according to the standard, are the self definition and neutral format that a VPF table presents. Primitive tables of another format in a VPF database do not meet the VPF standard.

The VPF standard will either need to be modified to include symbology or certain parts of the proposed VPFS (the CGM files) need to be moved outside of VPFS structure.

2.2 VPFS Strengths

The concept of uniform, cross product symbology has great potential for enhancing military inter-operability. Maps can be constructed from multiple VPF data products to support joint missions without fear of having the same map features on different products symbolized in different ways. Map users can view maps made by others without having to refer to a symbol key that varies from map to map.

VPFS will make it much easier to quickly create a quality map by eliminating the hand picking of symbols for each thematic query. The standard set of symbols will also allow the user to concentrate on map content rather than structure and layout saving valuable time.

VPFS enables feature level symbology. Prior to VPFS VPF symbology was assigned in an arbitrary manner by end user computer software applications and/or by end users. The assignment was at the query level, that is, all features resulting from a single theme query are symbolized in the same way. VPFS provides symbol condition tables which establish the rules relating symbols to feature code and attribute values. An example is feature code AA040 for a rig or superstructure. Symbol MP05002.cgm is used for rigs less that 46m high. Symbol MP05337.cgm is used for rigs greater than or equal to 46m in height. This allows application software to choose symbology on a feature by feature basis by comparing the feature attributes with the symbol attributes. As a further example, bathymetric contour lines could be assigned a different line color based on depth. A comparable effect can also be achieved within the view definition provided by the VPFVIEW software using query level symbology by creating multiple themes to retrieve features within bounded attribute ranges and assigning different symbology to each theme, but only by extensive user configuration. This is how the pre-defined views supplied with the prototype were obviously constructed.

VPFS can support the automated generation of symbol catalogs such as Nautical Chart 1, the symbology key to all DMA paper nautical charts.

VPFS represents a potentially significant component in an end user application system..

2.3 VPFS Weaknesses

Poor conformance- The issues of non-conformance with the VPF Specification must be resolved, either by graphic files outside the VPFS or by modifying the VPF Specification.

Poor performance- To symbolize a feature, an application must read a condition file, a description file, and the CGM file itself. The *cond* directory has 319 condition files which can be expected to average only about 400 bytes each in VPF format. The time to access these numerous files (one access per feature) as well as the 319 nearly identical table headers make for a system that is less than optimally configured for speed or size. Add to the overhead any index files such as the required variable length index for variable text fields and access time increases still more.

Unnecessarily complicated- There appears to be little if any benefit from separating symbol conditions from symbol descriptions, or from placing the conditions for individual attributes into separate files. One table for each feature class should be sufficient.

Symbol attributes are in the wrong place- They should not be separated from the conditions. The relationship between symbol condition and symbol attribute should be 1:1 even though the relationship between feature code and symbol condition may be many : 1.

Too many files- There is no obvious benefit from having a separate condition table for each feature code.

Lack of support for attribute based labeling- It did not appear to be possible to construct the digital equivalent of a printed nautical chart using the supplied software and data. For example, hydrographic soundings are stored in the Digital Nautical Chart (DNC) as point features with the depth values stored as attribute HDP (Hydrographic Depth). The evaluator could choose a sounding symbol from the library of point symbols and could then perform a spatial query on one of the soundings to display the depth, but there was no obvious mechanism to use the HDP attribute value as the symbol. The evaluator was similarly unable to show contour line depth values. This weakness can be overcome by a more robust symbology design that incorporates attribute labeling. To illustrate this point a digital chart was constructed using VPFVIEW, the VPF-based DNC database, and VPFS symbology. The digital representation (Figure 1) could not display the depth values available on the paper nautical chart (Figure 2). They could be obtained only by querying each sounding point individually.

Lack of support for defining the behavior of features under spatial query- VPFVIEW provides a default spatial query behavior that generates a text listing of attribute names and values. Other more graphical behaviors are also desirable. These might include displaying a picture of a point feature or drawing a graph. For example, to demonstrate the usefulness of VPF as a format for oceanographic data, a VPF database was prepared consisting of point features, each of which contained a series of oceanographic data associated with that point at various ocean depths. The data included temperature, salinity, and sound speed for a number of depths at the location of the point feature. A more desirable behavior under spatial query would be to graph the temperature, salinity, and sound speed against the depth.

2.4 VPFS Documents

The VPFS documents as presented on the CDROM were, for the most part, well written and concise. However they contained a number of inconsistencies and typographical omissions and errors as outlined below:

2.4.1 MIL-STD-2412

Section 4.5. - The purpose and usefulness of the icon identifier is unclear.

Table 10 (page 32) - The column "LWeight" should have a data type of F,1 rather than the integer type it currently has.

Table 11 (page 33) - The column names "Types" and "Text" are transposed.

Table 15 (page 36) - The column specification "Security_Class" should indicate a data type of T,1. It currently has no length indicator.

Table 13 (page 34) - The text row ID column (TROWID) should be specified as data type I,1 not I,12. The icon ID column (ICON_ID) should be specified as T,12.

Figure 1. Norfolk Approach
created using DNC01.

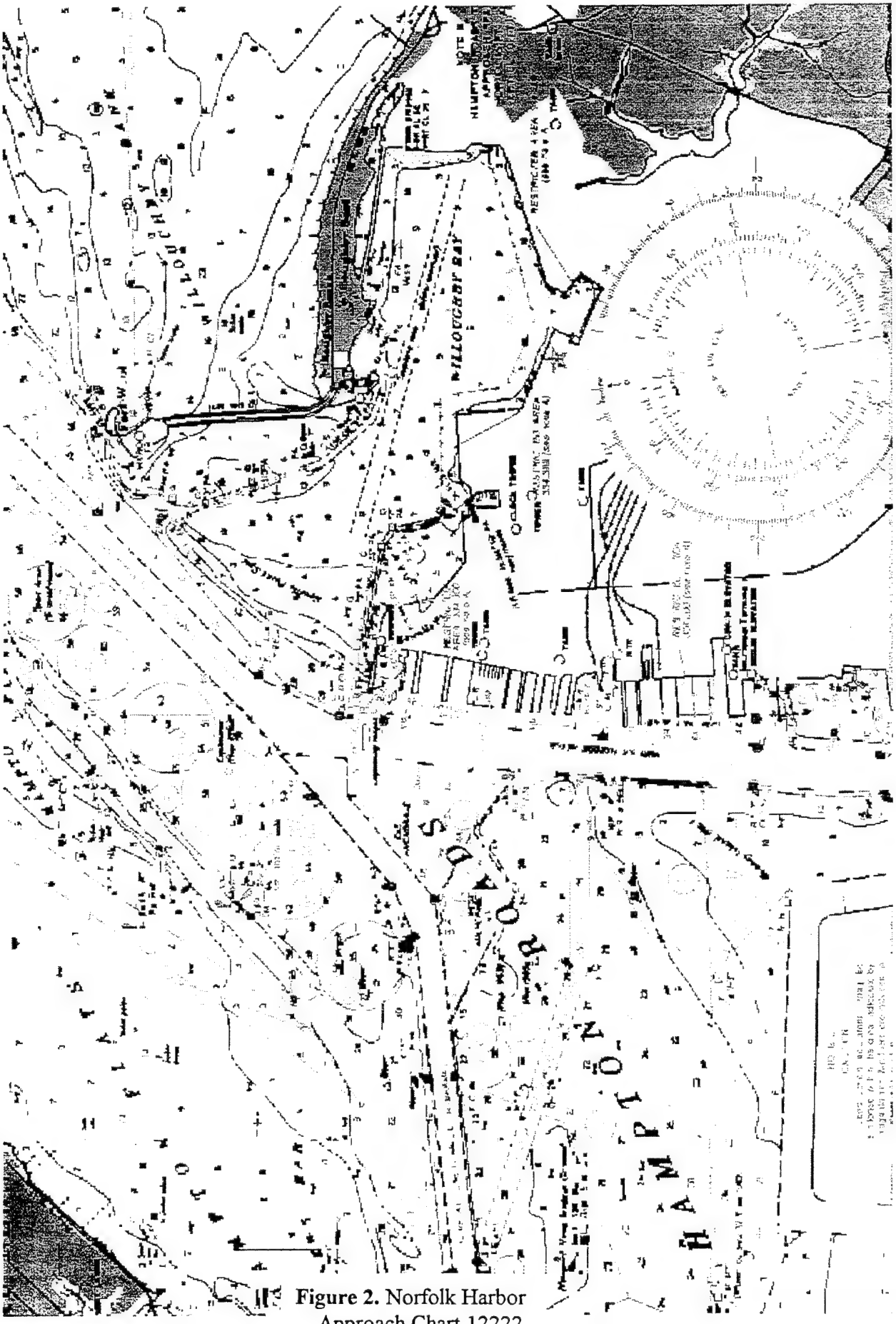


Figure 2. Norfolk Harbor Approach Chart 12222

Table 16 (page 38) - The logical condition syntax should more closely follow Structured Query Language (SQL) rules and their representation in this table should be exactly as the database developer should use them. An example of the discrepancies in representation are the greater than and less than symbols that cannot be coded in standard ASCII as displayed in Table 16. An example of nonstandard operators is the negation symbol that should be represented as "<>" to be more SQL-like.

2.4.2 MIL-V-89045

The table of contents should be updated to match the page layout of the specification.

Table 3 Database Header Table - This table is specified differently than that of MIL-STD-2412 in both content and data types. A corrected version is shown below.

(Header Length)L;
Symbol Database Header Table;-;
ID=I,1,P,Row Identifier,-,-,-,:
VPFS_VERSION=T,10,N,VPF Version number,-,-,-,:
DATABASE_NAME=T,8,N,Directory name of this database,-,-,-,:
DATABASE_DESC=T,100,N,Description of this database,-,-,-,:
MEDIA_STANDARD=T,20,N,Media Standard,-,-,-,:
ORIGINATOR=T,50,N,Producer of this database,-,-,-,:
ADDRESSEE=T,100,N,Address of the producer,-,-,-,:
MEDIA_VOLUMES=T,1,N,Number of Volumes in this database,-,-,-,:
SEQ_NUMBERS=T,1,N,The Sequential Number(s) in this database,-,-,-,:
NUM_DATA_SETS=T,1,N,Number of Data Sets,-,-,-,:
SECURITY_CLASS=T,1,N,Security classification,-,-,-,:
DOWNGRADING=T,3,N,Downgrading,-,-,-,:
DOWNGRADE_DATE=D,1,N,Date,-,-,-,:
RELEASABILITY=T,20,N,Releasability restrictions of data,-,-,-,:
TRANSMITTAL_ID=T,1,N,Unique Transmittal Identifier ,-,-,-,:
EDITION_NUMBER=T,10,N,Edition Number of this database,-,-,-,:
EDITION_DATE=D,1,N,Date of edition,-,-,-,:
SUP_PROD_SPEC=T,80,N,VPF product specification supported by this product,-,-,-,:
PROD_SPEC1_DATE=D,1,N,Date of supported VPF product specification,-,-,-,:;

To remain consistent with section 3.3.7 that specifies that DOS filename conventions are to be used, column definitions that contain filenames should be specified as T,12 - a space large enough to contain a DOS filename.

The hierarchy of symbol attributes remains undefined. For instance, line symbols have color, weight, and pattern attributes both in the description tables and within the CGM files themselves. Do the attributes in a VPFS symbol description table override similar attributes in the CGM file?

2.4.3 MIL-HDBK-857

Section 1 explains the goals and approach to the VPFS development effort. While many different arguable positions may be taken with respect to symbology goals, the stated goal of achieving a cross-product, map analysis oriented, and extensible symbology is probably the most useful in the planned migration from a product orientation to an information service orientation.

Section 1.2.5.1 describes plans to save storage by distributing symbols in CGM binary format only. Since the entire prototype symbol collection requires less than 1 megabyte of storage in either form, both the clear text and the binary forms should be maintained.

Section 2.3.1 Symbology Descriptions- The symbol description file format is not addressed. The actual files are comma delimited ASCII text. If VPFS is to be constructed as a VPF database, then these should be VPF tables, one for each point, line, area, and text symbol. F_CODE should be the primary key.

Section 2.3.2 Symbology Conditions- The symbol condition file form is described as delimited ASCII. If VPFS is to be constructed as a VPF database, then these should be VPF tables, perhaps one for each for point, line, area, and text symbology.

Section 4.1 Symbology Database- VPFS is described as "using VPF for the data model upon which symbology can be maintained."

The latest documents indicate that the final VPF symbology database will implement binary CGM files. What are the implications for the VPFVIEW software which now uses the clear text symbols? Are the binary files operable on all computers?

2.5 The CD-ROM Symbology Database Prototype

The symbology prototype offered on CD differs substantially from the VPF symbology standard and product specification. Some specific instances follow:

The ASCII symbol condition table representations in the *con* directory do not represent the format outlined in MIL-STD-2412. They contain columns such as the product identifiers (DNC, VMAPL0, etc.) that, according to the VPFS specification should not be present, and columns such as the icon ID (ICON_ID) that should be there but are absent.

The description tables in the *desc* directory contain column definition names that are different than what is specified in MIL-STD-2412 and in the case of the point description table, a extra column (F_CODE) has been added.

MIL-STD-2407 (VPF) appears to require a TABLE column in a value description table. MIL-STD-2412 defines no such column. This column relates the entry back to an attribute table or, with a null entry, signals that the description applies to all attribute tables which reference it.

Several rows in the line description table (linepatt.vdt) have descriptions which exceed the 40 character column field width. These include rows 8, 12, 18, and 20. Either the field width should be increased or the descriptions modified to permit the descriptions to fit the allotted space.

The syntax of the logical symbol conditions in the symbol condition tables do not match what is specified in MIL-STD-2412 and are not consistent across all condition files. Examples include the logical OR written as "or" in file ZD045.con and as "||" in file ZD040.con.

There was not the expected 1:1 relationship between the clear text CGM files and the binary CGM files. There were 536 binary files but only 419 clear text files.

3. NRL Proposed VPFS Design

The 319 separate condition tables present a formidable performance problem to direct use of VPFS. The association of symbol attributes with symbol definition seems meaningless and is likely redundant or in conflict with symbol attributes within the CGM files.

An alternate design such as the combination of the symbol condition tables with the symbol description tables should be considered (Figure 3). This increases the number of rows slightly (a many:1 relationship), but it reduces the number of required tables from 323 (319 condition tables and 4 description tables) to 4 (1 for each geometric primitive). These combined tables would have a row for each symbol condition. The entire condition statement would be contained in a single standard SQL where clause, rather than multiple simple expressions.

Using this design, symbol color and style attributes become associated with symbol condition rather than symbol definition, which is more intuitive. This change would simplify and accelerate the symbol lookup, improve the power of VPFS by associating symbol attributes with symbol conditions, and encourage direct usage of VPF.

An example and comparison of operations involving the current VPFS table scheme and this proposed design are given in Appendix A.

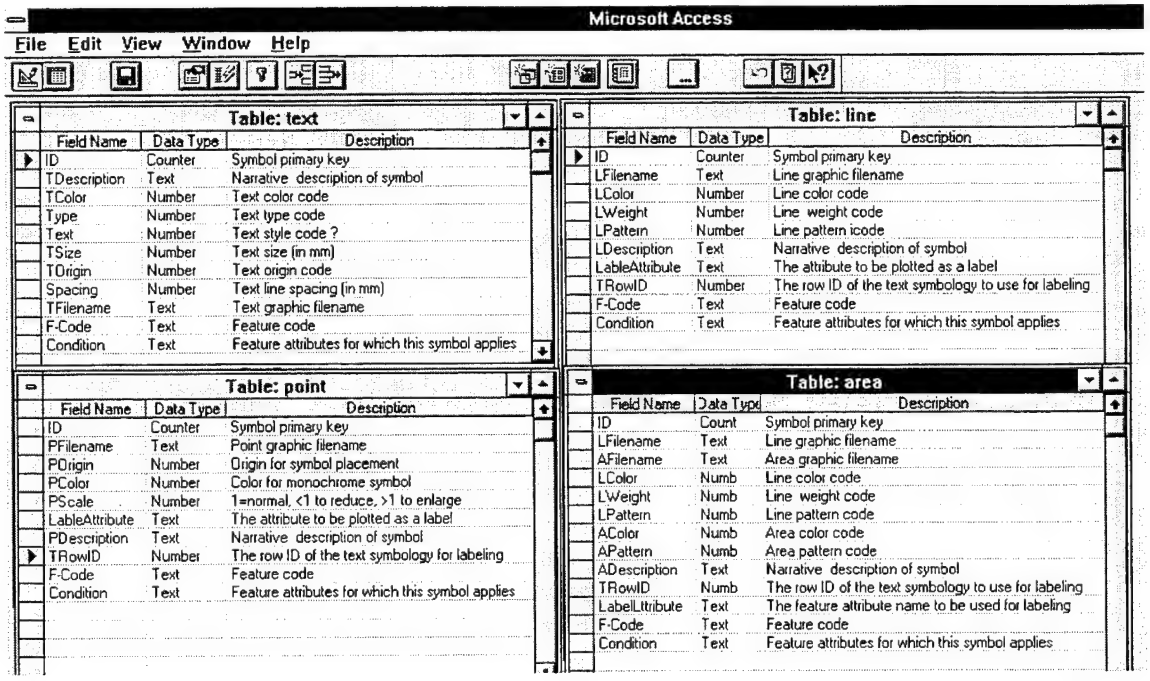


Figure 3. Proposed VPFS symbol conditions table definitions for point, line, text and area.

3.1 Annotating Point Features with an Attribute Value.

This annotation will be in addition to the current point symbology and will allow elevation values to be plotted at spot elevation points. Add additional columns, Label_Attribute and TRowID, to the point symbol table. Label_Attribute provides the name of the attribute whose values are to be used to annotate the point feature. TRowID identifies the row in the text symbol table that will provide the text symbol attributes. To symbolize a point feature in this manner:

1) Find all occurrences of the feature's feature code in the point feature table. If more than one is found, compare the feature's attribute values with the condition expressions until a match is found.

2) Call functions to set the drawing modes to correspond to the point symbol attribute values.

3) Draw the point symbol.

4) Follow the TRowID into the text symbology table to get the text attributes.

5) Call functions to set the text drawing attributes.

6) Convert the attribute value to text if it is numeric.

7) Draw the text.

In an entirely analogous manner, it is also desirable to annotate line features and area features with an attribute value. This will provide a means of labeling contour lines and areas with the corresponding elevation value.

3.2 Symbolizing Behavior Under Spatial Query

In VPFVIEW spatial queries are performed by clicking on a displayed feature to provide a text listing of attribute names and values. Other more graphical behaviors under spatial query are also desirable. These might include displaying a picture of a point feature or drawing a graph.

Add an additional column SQBehavior of type text to the point symbol table. The text column SQBehavior will contain a human readable but parsable description of the desired behavior when the features is queried. Sample behaviors might be:

SQBehavior = Default to produce a text listing of the attribute names and values.

SQBehavior = Picture BRIDGE4259384.TIF to display the image contained in the indicated file.

SQBehavior = Graph to graph numerically code attribute names and values.

SQBehavior would be parsed by application software which would invoke functions to perform the desired action.

4. Conclusions and Recommendations

1. Although point symbology is a vector data type, it does not fit the VPF format as evidenced by the decision to place individual symbols in CGM format files. Furthermore, line, area, and text symbology do not fit very well into CGM either, as evidenced by the continued use of symbol attributes in the symbol definition files. We recommend that graphics files (CGM and XBM) should be located outside of the VPFS database. The names of these graphics files should, however appear in the symbol tables inside the VPFS database. The CGM format should be used to store point symbols. The XBM format should be used to store area and line patterns.

2. The VPFS Standard defines the symbology tables to be VPF tables, but in the VPFS prototype, they are ASCII using various delimiters including comma, semicolon, and space. We recommend that the symbology tables be VPF tables as specified by the VPF Standard and that a software tool be provided to translate VPF tables to and from common RDBMS data exchange formats such as XBASE (a common personal computer format) and delimited ASCII (comma and tab).

3. VPFS and VPFVIEW provide no mechanism to symbolize attribute values as text associated with point, line, and area features. For example, hydrographic soundings normally appear on nautical charts, but there is no way to display them using VPFS and VPFVIEW. Similarly, there is no way to label contour lines. We recommend that both VPFS and VPFVIEW be expanded to support this attribute labeling.

4. VPFS does not address the issue of behavior under spatial query. A mechanism for doing this is suggested in Section 3.2 that supports the display of both stored pictures and generated graphs.

5. VPFS does not support the use of the same point symbol in different colors as it does for line, area, and text symbols. Color would be a very useful way to symbolize point feature attribute values such as sounding depth, just as color is a useful way to symbolize line (contours for example) and area (land/water areas for example) features. Point symbols could be considered as two groups, a monochrome group and a color group. The monochrome symbols contain no color information within the CGM file and should rely on the color being set based on the symbol's color attribute. The color symbols contain multiple colors and so must have color defined within the CGM file and should not have a color attribute. We recommend that VPFS be expanded to include the point symbology attribute of color.

6. There will be a slight performance penalty associated with the use of the feature level symbology supported by VPFS. Application software designers should carefully weigh the benefits of applying feature level symbology against the additional overhead of looking up symbology for each feature instead of for each query.

7. The VPFS symbol condition tables use a separate column for each symbol condition. This may have been the reason for having a table for each feature code. We recommend that these multiple condition clauses be combined into a single standard compound SQL "where" clause.

8. An alternate design was proposed in Section 4 of this evaluation. In the alternatative design, we proposed to greatly simplify the VPFS design and improve its performance by merging the 319 condition tables into the 4 symbol description tables. The alternative design supports most of the recommendations made above including attribute labeling of point, line, and area features and symbolizing point feature attributes by point symbol color. It also addresses behavior under spatial query. We recommend that the alternative design be considered and that a technical exchange meeting be held to further refine the VPFS design.

5. Acknowledgments

This effort was sponsored by the Digital Mapping Charting and Geodesy Analysis Program (DMAP), funded by the Oceanographer of the Navy under Program Element 0603704N through TOWS program managers Mr. Ken Ferer and Mr. Harry Selsor.

Technical review of this report was provided by Mr. Mike Harris and Ms. Maria Kalcic, both of the NRL Mapping, Charting, and Geodesy Branch, and Ms. Mary Clawson of the NRL Marine Geosciences Division.

Appendix A - VPF Symbol Database Access Example

This example traces the steps required to identify and access a feature symbol from a VPFS database as defined in MIL-STD-2412 (part A) and then in part B, using the NRL proposed design.

Part A

1. Feature Selection

The feature we wish to match to a symbol is located at row 12 in the INDUSTP.PFT point feature table in a VPF database. It is a flare pipe standing 48 meters above the surface level. Below is the data from row 12 in the INDUSTP.PFT point feature table.

Row ID: 12
FACC Code: AF070
Accuracy Category: 1
Conspicuous Category: 4
Height Above Surface Level: 48
Location Category: (Null)
Name: N/A
Product Category: (Null)
Usage: 41
Highest Z-value: (Null)
Tile Reference Identifier: 2
Entity Node Primitive Foreign Key: 1

---

2. Comparing the Logical Conditions in the Condition Table

To locate a symbol, the feature code is used to locate the condition file in the *COND* directory of the symbol database. This particular condition table has two rows that use the attribute HGT (height above surface) to select the appropriate symbol. The HGT attribute of the feature we wish to symbolize is 48 meters and, using the logical condition description in column 2 of the table, the second row is selected as having the applicable attribute values. Reviewing the row ID entries, it is then found that this symbol's description resides at row 337 of the point description table. The two rows from the AF070.CON symbol condition table are presented below.

Row ID: 1
HGT: < 46m
Area description row ID: 0
Line description row ID: 0
Point description row ID: 20
Text description row ID: 0
ICON identifier: MPAF07005020

Row ID: 2
HGT: >= 46m
Area description row ID: 0
Line description row ID: 0
Point description row ID: 337
Text description row ID: 0
ICON identifier: MPAF07005337

3. Accessing the Symbol Description Table

The next step is to locate row 337 of the POINT symbol description table in the *DESC* directory. This table provides a description of the symbol, its origin code (used for symbol placement) and the CGM file in which the actual symbol is located. Below is row 337 of the POINT symbol description table.

Row ID: 337
Point graphic filename: MP05337.cgm
Origin for symbol placement: 5
Narrative description of symbol: black vertical obstruction symbol

---

4. Retrieving the Symbol

From the information in the symbol description table, the CGM symbol file can be located and read. These tables are located in the *GRAPHICS* directory of a VPFS database. Below is the clear-text version of the CGM symbol file MP05337.CGM.

BegMF "CorelDRAW! for UNIX, PRIOR Data Sciences Product Sales Inc.";
MFVersion 1;
MFDesc "CorelDRAW! Clear Text CGM version 1.00";
VDCType integer;
ColrPrec 255;
ColrValueExt 0 0 0 255 255 255;
MFElemList " BegMF EndMF BegPic BegPicBody EndPic MFVersion MFDesc VDCType
ColrPrec ColrValueExt MFElemList FontList ScaleMode ColrMode
LineWidthMode EdgeWidthMode VDCExt BackColr AuxColr Transparency Line
IncrLine Polygon IncrPolygon Rect Circle Ellipse EllipArc EllipArcClose
LineType LineWidth LineColr IntStyle FillColr EdgeType EdgeWidth
EdgeColr EdgeVis ";
BegPic "Picture 1";
ScaleMode abstract 0.0;
ColrMode direct;
LineWidthMode abstract;
EdgeWidthMode abstract;
VDCExt (-100, -156) (100, 156);
BackColr 255 255 255;
BegPicBody;
IntStyle solid;
FillColr 0 0 0;
EdgeType 1;
EdgeWidth 14;
EdgeColr 0 0 0;
EdgeVis off;
IncrPolygon (-100, -118) (36, 74) (64, 200) (64, -200) (36, -74) (-39, 18)
(-61, 72) (-61, -72);
IncrPolygon (32, -128) (-1, 1) (0, 3) (0, 3) (-1, 2) (-1, 3) (-2, 3)
(-1, 2) (-2, 2) (-2, 2) (-2, 2) (-2, 1) (-3, 2) (-2, 1) (-3, 1) (-2, 0)
(-4, 1) (-2, -1) (-3, 0) (-2, 0) (-3, -1) (-2, -1) (-3, -2) (-2, -1)
(-2, -2) (-2, -2) (-2, -2) (-1, -2) (-2, -3) (-1, -3) (-1, -2) (0, -3)
(0, -4) (0, -2) (0, -3) (0, -2) (1, -3) (1, -2) (2, -3) (1, -2) (2, -2)

(2, -2) (2, -2) (2, -1) (3, -2) (2, -1) (3, -1) (2, 0) (5, 0) (1, 0)
(3, 0) (2, 0) (3, 1) (2, 1) (3, 2) (2, 1) (2, 2) (2, 2) (2, 2) (1, 2)
(2, 3) (1, 2) (1, 3) (0, 2) (1, 5);
EndPic;
EndMF;

Part B

1. Feature Selection

Again the feature we wish to match to a symbol is located at row 12 in the INDUSTP.PFT point feature table in a VPF database.

Row ID: 12
FACC Code: AF070
Accuracy Category: 1
Conspicuous Category: 4
Height Above Surface Level: 48
Location Category: (Null)
Name: N/A
Product Category: (Null)
Usage: 41
Highest Z-value: (Null)
Tile Reference Identifier: 2
Entity Node Primitive Foreign Key: 1

---

2. Comparing the Logical Conditions and Locating the CGM Symbol

This time, using the NRL-proposed design, the single point condition table is accessed. Each point feature code occupies a row for each set of logical conditions. In this case, the feature code AF070 is contained in two rows (it has a single set of logical conditions). The *Condition* field is parsed and compared to the feature attribute table to identify the applicable symbol. Symbol attribute information is also immediately available such as scale and origin that can be accessed if needed. The PFilename field of row 12 (the appropriate symbol for these conditions) records the CGM filename. The *label attribute name* field provides a method of plotting a point's attributes as text. In this case, the name (attribute *NAM*) of the feature could be displayed alongside or in place of the point symbol. The two rows in the point condition table that contain feature code AF070 information are displayed below.

Row ID: 11
Feature Code: AF070
Point graphic filename: MP05020.cgm
Origin for symbol placement: 5
Symbol Color: 0
Symbol Scale Code: 1
Label Attribute Name: NAM
Narrative description of symbol: black vertical obstruction symbol
Text Row ID: 7

Condition String: HGT < 46m

Row ID: 12
Feature Code: AF070
Point graphic filename: MP05337.cgm
Origin for symbol placement: 5
Symbol Color: 0
Symbol Scale Code: 1
Label Attribute Name: NAM
Narrative description of symbol: black vertical obstruction symbol
Text Row ID: 7
Condition String: HGT >= 46m

---

3. Retrieving the Symbol

The same symbol is retrieved in a manner identical to that outlined in part A.